Étude préliminaire d'une méthode de prise de décision adaptative pour la commande vocale dans un habitat intelligent

Alexis Brenon, François Portet, Michel Vacher

Étude préliminaire d'une méthode de prise de décision adaptative pour la commande vocale dans un habitat intelligent

A. Brenon[1,*] F. Portet[1] M. Vacher[2]

[1] Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
[2] CNRS, LIG, F-38000 Grenoble, France
[*] Jeune chercheur (doctorant)

{prenom.nom}@imag.fr

Résumé

Dans les habitats intelligents, les prédictions et décisions qui sont souvent faites a priori nécessitent de la part de l'utilisateur une configuration qui peut être complexe et fastidieuse. Ces habitats ont pourtant des capacités de perception requises pour analyser le comportement de l'utilisateur et modifier ses décisions automatiquement. Nous présentons une étude préliminaire qui teste un système de décision à partir d'une commande vocale et de son contexte modifié par renforcement. Le système expérimenté sur un corpus réaliste montre le potentiel d'une telle adaptation.

Mots Clef

habitat intelligent, informatique sensible au contexte, intelligence ambiante, apprentissage par renforcement

Abstract

In smart homes, prediction and decision are often defined a priori and require tuning from the user, which can be tedious, and complex. However, these smart home can demonstrate the ability to analyse the user behavior and so as to modify its decisions automatically. We present a preliminary study that tests a decision system from voice command and context which is modified by reinforcement learning. The system was run on a realistic corpus which shows the interest of such an adaptation.

Keywords

smart home, context-aware computing, Ambient Intelligence, reinforcement leranring

1 Introduction

Dans le domaine de l'intelligence ambiante (AMI), l'habitat intelligent (aussi appelé *Smart Home*) a vu le jour pour augmenter l'expérience et le contrôle des utilisateurs et fournir un soutien aux personnes en perte d'autonomie. Pour fournir ce contrôle enrichi, ces systèmes perçoivent leur environnement et décident des actions à appliquer à celui-ci pour le modifier. Cette décision peut être prise soit de manière réactive après certains événements spécifiques tels que la demande d'un utilisateur, soit de façon programmée (p. ex. : tous les vendredi) ou bien de manière proactive par exemple, en prédiction d'une situation à risque. Cette perception est non seulement utile pour déclencher une décision, mais aussi pour adapter la décision aux circonstances dans lesquelles une telle décision doit être exécutée. Dans le domaine AMI, ces circonstances sont appelées le *contexte* et les systèmes, qui prennent explicitement en compte le contexte sont dits *context-aware* (sensible au contexte).

Pour illustrer le rôle du contexte dans la prise de décision, prenons l'exemple d'une maison intelligente contrôlée par la voix qui connaît un regain d'intérêt dans la communauté cette décennie [14, 1, 12, 7, 9, 6]. Ce type de contrôle repose sur une interface vocale (VUI – *Voice User Interface*) qui permet une communication 'naturelle' avec le système et qui est particulièrement bien adaptée aux personnes à mobilité réduite et à des situations d'urgence (mains-libres et interaction à distance) [22, 24]. Dans un tel cadre, une commande vocale peut être 'allumer la lumière', 'vérifier la porte', 'appeler ma fille', etc. Dans ce cas, le contexte est utile pour lever l'ambiguïté de la commande afin de prendre la décision adéquate. En effet, dans un énoncé tel que 'allumer la lumière' prononcé dans une chambre avec plusieurs lampes, l'utilisateur n'énumère généralement pas les lampes qui doivent être allumées car cet utilisateur attend de son interlocuteur qu'il les devine correctement [24]. De plus, il ne serait pas naturel de demander à l'utilisateur de spécifier tous les détails d'une commande, c'est au système de prise de décision d'évaluer le contexte pour prendre la décision la plus adéquate.

L'état de l'art présente un certain nombre d'approches ayant mis en œuvre un système de décision dans un habitat intelligent. Par exemple Moore et coll. [19] ont mis au point un système qui exploite un ensemble de règles floues afin de trouver l'action la plus appropriée dans un contexte donné. On peut également citer Kofler et coll. [15] et Gómez-Romero et coll. [10] qui utilisent la logique de description pour définir le comportement d'un système

sensible au contexte ou Leong et coll. [17] ainsi que Yau et Liu [27] qui modélisent le comportement d'un système perceptif par des règles ECA (événement-condition-action). D'autres approches basées sur les réseaux bayésiens ont été mises en œuvre [16, 18, 21, 2, 3] pour prendre en compte l'incertitude sur les données. Cependant, dans ces propositions, le système ne s'adapte qu'à la situation courante et ne prend pas en compte les évolutions d'utilisation qui peuvent intervenir. En effet, selon les saisons, l'heure de la journée, la survenue d'un invité ou suite à un handicap passager ou permanent, les habitudes et contextes d'utilisation peuvent changer et nécessiter un reparamétrage. De plus, ces systèmes requièrent un paramétrage de l'utilisateur pour prendre en compte ses préférences, paramétrage pouvant être parfois très complexe [23]. Par ailleurs, certaines erreurs d'interprétation d'une commande vocale ou d'un contexte pourraient être identifiées par le système qui pourrait les corriger automatiquement afin d'éviter l'appel à une paramétrisation fine et complexe. Ces contraintes militent pour un système souple et évolutif capable de s'adapter au comportement parfois changeant des individus dont il est censé simplifier la vie.

Ce fut le cas dans le projet ACHE (*Adaptive Control of Home Environment*) [20] dans lequel un système de contrôle de la domotique n'est pas explicitement programmé par l'utilisateur mais appris par l'observation du comportement de l'utilisateur face à des décisions prises par le système. Les actions sont prédites par un réseau de neurones et la réaction de l'utilisateur est ensuite analysée. Si l'utilisateur corrige l'action alors l'utilité de cette action est modifiée en utilisant des techniques d'apprentissage par renforcement. Cette architecture a été mise en œuvre pour des tests sur le contrôle de la lumière dans un habitat intelligent, cependant ce projet ne semble pas avoir été mené à son terme et n'utilise pas explicitement le contexte pour prendre sa décision.

Dans cet article, nous présentons une première étude qui teste la capacité d'adaptation d'une technique d'apprentissage par renforcement sur des données acquises dans le cadre d'une expérimentation dans un habitat intelligent dont le but est de développer un système s'adaptant à l'utilisateur et à l'environnement sur le long terme. L'article présente en section 2 la méthode du Q-learning et comment celle-ci est particulièrement bien adaptée au problème de décision suite à une commande vocale. La section 3 décrit l'expérimentation de cette méthode sur un corpus acquis en condition réaliste. L'article se termine par une discussion des résultats et des perspectives d'améliorations.

2 Méthode

La méthode que nous proposons reposant sur le Q-learning, nous allons faire un bref rappel de cette technique et introduire comment celle-ci a été mise en œuvre dans le cas d'un système de décision pour un habitat intelligent.

2.1 Apprentissage par renforcement & Q-learning

L'apprentissage par renforcement est une technique d'apprentissage automatique qui permet à un agent d'apprendre son comportement grâce à un retour de l'environnement sur lequel il agit. Ainsi, un problème d'apprentissage par renforcement repose sur trois principales composantes [13] :

- L'**environnement**, qui change d'*état* lorsque le système exerce une *action* sur lui ;
- La **fonction de renforcement** qui définit l'objectif du système en attribuant une récompense (positive, négative ou nulle) à chaque paire état-action ;
- La **fonction de valeur** qui associe à chaque état du système une valeur correspondant à l'ensemble des récompenses reçues à partir de cet état jusqu'à un état final.

L'objectif de l'agent est alors de déterminer une stratégie, basée sur la fonction de valeur, par une suite d'interactions essai-et-erreur avec l'environnement. Les premières approches utilisées se basaient sur les principes de la programmation dynamique de manière à modifier la fonction de valeur (généralement représentée sous la forme d'une table d'association) jusqu'à convergence.
En 1989, Watkins publie une extension aux approches classiques de programmation dynamique qu'il nommera le Q-*Learning* [26]. Il introduit alors la notion de Q-*Value* associée non plus à un état du système mais à une paire état-action. De même, la notion de fonction de valeur est remplacée par celle de Q-fonction ou de fonction de Q-valeur. Il définit également l'équation permettant de mettre à jour les valeurs de la Q-fonction, en respectant la forme de l'équation de Bellman largement utilisée pour la programmation dynamique.

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \qquad (1)$$

où s_t désigne un état de l'environnement au moment t, a_t une action au moment t, $r(s_t, a_t)$ la récompense reçue pour l'action a_t, γ le facteur d'actualisation et $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ la valeur de Q pour la meilleure action a_{t+1} dans l'état suivant s_{t+1}.
Cette nouvelle approche permet de limiter le coût de calcul de l'apprentissage par renforcement dans le cas d'environnements non déterministes. En effet, alors que l'utilisation de la programmation dynamique nécessite le calcul d'une sommation sur un nombre théoriquement infini d'interactions, le Q-Learning ne nécessite que les valeurs de l'état courant et suivant (ou courant et précédent) pour mettre à jour la Q-fonction. Toutefois, après un certain nombre d'interactions, la Q-valeur associée à un état est assurée de converger vers sa valeur optimale comme c'était le cas dans les approches précédentes.
Dans les cas les plus simples, la fonction de Q-valeur se résume à une matrice permettant d'associer une valeur à chaque paire état-action. L'implémentation la plus simple du Q-Learning consiste alors à mettre à jour la matrice

après chaque interaction en suivant l'équation suivante reprenant l'équation 1 :

$$Q^{t+1}_{s_t,a_t} = Q^t_{s_t,a_t} + \alpha \left(r(s_t, a_t) + \gamma \max_{a_{t+1}} Q^t_{s_{t+1},a_{t+1}} - Q^t_{s_t,a_t} \right) \quad (2)$$

où Q^t représente la matrice actuelle, Q^{t+1} la matrice de valeur après mise à jour et α le facteur d'apprentissage qui permet de faire varier la vitesse d'apprentissage de l'agent.

2.2 Habitat intelligent commandé par la voix

Un habitat intelligent est un exemple d'application de l'intelligence ambiante. Il s'agit d'un logement qui intègre des capteurs et des actionneurs (issus de la domotique) et qui est capable de *percevoir* l'environnement pour *agir* sur celui-ci de manière réactive ou pro-active [5].
Une application de commande vocale illustre bien cette nécessité de perception et d'action. Par exemple, si un utilisateur dans un habitat intelligent prononce "Allume la lumière", une autre personne n'aura certainement aucune difficulté à interpréter cette demande et à agir. Un système informatique, s'il veut agir de la manière la plus naturelle possible, doit, par contre, inférer seul la lampe qui est implicite dans l'énoncé (dans le cas où il y aurait plusieurs lampes) ainsi que son intensité (dans le cas ou celle-ci serait réglable). Cette information manquante doit être récupérée à partir de la connaissance du contexte : le système doit déduire l'emplacement de l'utilisateur pour allumer la lumière dans la pièce où le participant se trouve et il doit également déduire de son activité le lieu et/ou l'intensité de l'éclairage. Si nous prenons comme exemple une personne qui vient juste de se réveiller dans une chambre à coucher, la lampe de chevet à faible puissance pourrait être plus appropriée que le plafonnier à pleine puissance. Dans cet exemple, le contexte est constitué de deux paramètres déduits à partir de données de capteurs : la localisation et l'activité.
Dans notre étude, les actions que nous considérerons possibles sont les suivantes :

- allumer/éteindre la {lumière, radio}
- ouvrir/fermer les {stores, rideaux}
- donner la {température, heure}
- demander un appel visio/téléphonique ou un appel d'urgence.

Ces actions constituent un sous-ensemble d'actions possibles définies suite à une étude utilisateurs [22]. Bien sûr, cet ensemble d'actions doit être adapté à chaque utilisateur et logement, mais cette liste prédéfinie a été utile pour l'évaluation du système. Cette étude a également permis de définir des commandes vocales en utilisant une grammaire très simple comme le montre la figure 1. Chaque commande commence par un mot-clé unique qui permet de savoir si la personne s'adresse au système ou non. Dans ce qui suit, nous allons utiliser *'Nestor'* comme mot-clé.
Ce type d'environnement semble particulièrement bien se prêter au Q-learning. En effet, l'ensemble d'états, quoique potentiellement grand, est fini et discret et l'ensemble des actions est également fini et discret.

```
basicCmd          = key initiateCommand object |
                    key emergencyCommand
key               = "Nestor"
initiateCommand   = "ouvre" | "ferme" | "baisse" | "éteins" | "monte" |
                    "allume" | "descend" | "appelle" " | "donne"
emergencyCommand = "au secours" | "à l'aide"
object            = [determiner] ( device | person | organisation)
determiner        = "mon" | "ma" | "l'" | "le" | "la" | "les" | "un" |
                    "des" | "du"
device            =  "lumière" | "store" | "rideau" | "télé" | "télévision" |
                     "radio" | "heure" | "température"
person            = "fille" | "fils" | "femme" | "mari" | "infirmière" |
                    "médecin" | "docteur"
organisation      = "samu" | "secours" | "pompiers" | "supérette" | "supermarché"
```

FIGURE 1 – Extrait de la grammaire des commandes vocales

3 Expérimentation

Le Q-learning a été mis en œuvre et testé à partir de données réalistes. Cette section présente les scénarios réalisés et les données recueillies lors d'une étude précédente, détaille la manière dont l'apprentissage a été effectué ainsi que les résultats.

3.1 Scénario d'utilisation de la maison intelligente contrôlée par la voix

L'habitat intelligent considéré dans cette étude est l'appartement DOMUS conçu par le *Laboratoire d'informatique de Grenoble* (LIG) [8]. La figure 2 illustre la configuration de l'appartement. Il s'agit d'un logement de 30 mètres carrés comprenant une salle de bains, une cuisine, une chambre et un bureau. Toutes ces pièces sont équipées de capteurs et d'actionneurs tels que des détecteurs de mouvement infrarouges, des capteurs de contact, des caméras vidéo (utilisées uniquement à des fins d'annotation), etc. En outre, sept microphones ont été placés dans le plafond pour la capture audio. L'appartement est entièrement utilisable et peut accueillir un habitant pendant plusieurs jours. Plus de 150 capteurs sont gérés dans l'appartement pour fournir différents services (p. ex. : l'éclairage, l'ouverture/fermeture des volets, la gestion média, etc.).

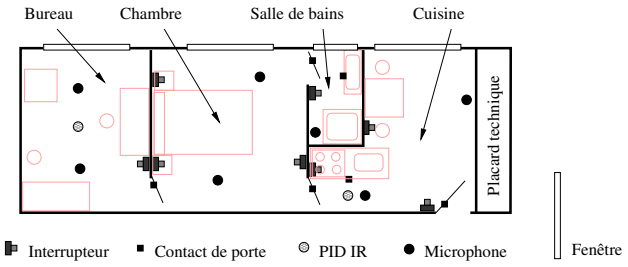

FIGURE 2 – Plan de l'appartement DOMUS et disposition des capteurs

Afin de recueillir des données de décisions en contexte, plusieurs participants ont été recrutés pour jouer des scénarios de la vie courante dans l'appartement. Il a été notamment demandé aux participants de prononcer des commandes vocales pour activer les actionneurs dans la maison intelligente. L'objectif de cette expérience était de tester un contrôleur intelligent [4] en fonctionnement réel dans

les situations de vie correspondant aux commandes vocales prononcées par l'utilisateur. Les situations que nous avons considérées dans l'étude étaient les suivantes : (1) faire le ménage dans l'appartement, (2) préparer et consommer un repas, (3) converser par vidéoconférence, (4) avoir des activités de loisirs (lecture), (5) faire une sieste. Afin de guider les participants lors de la réalisation de l'expérience, la grammaire des commandes vocales a été fournie (cf. figure 1) ainsi qu'un scénario de scènes de vie courante. Ce scénario a été conçu pour durer environ 45 minutes, cependant il n'y avait pas de contrainte sur le temps d'exécution. Les quatre premières parties mettaient l'utilisateur en situations d'activités quotidiennes tout en prononçant des commandes vocales. La figure 3 montre la première partie du scénario. Chaque participant reçoit une liste d'actions à effectuer et les commandes vocales à prononcer. La dernière partie du scénario était réservée à une interaction vocale avec l'extérieur (visio-conférence) et ne sera pas utilisée dans cette étude. Chaque participant devait utiliser une commande vocale pour activer une commande (allumer/éteindre les lumières, ouvrir/fermer les stores, etc.). La consigne était de répéter cette commande jusqu'à 3 fois en cas de défaillance du système. En cas de défaillance persistante un magicien d'Oz était utilisé pour faire croire à une activation correcte.

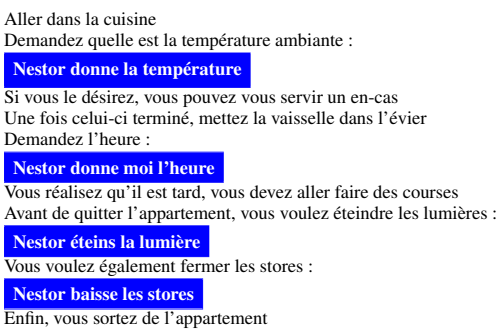

FIGURE 3 – Exemple de scenario dont la réalisation était demandée aux participants

Au total, 15 personnes (9 femmes, 6 hommes) ont participé à l'expérience. L'âge moyen des participants était de 38 ans $\pm 13,6$ (19-62 ans, min-max). 11 heures de données ont été enregistrées. Toutes les expériences ont été filmées mais seulement à des fins d'annotation. Les vidéos ne font donc pas partie du corpus. Pour l'étude présentée dans cet article, le corpus de commande domotique est donc composé d'un ensemble de situations (activité de la personne, lieu, état de l'appartement), d'énoncés de commande vocale ainsi que les actions domotiques correspondant à ces commandes. Ce jeu de données est extrait du corpus Sweet-Home qui est présenté en détail dans [25].

3.2 Apprentissage

L'apprentissage de notre modèle a été fait sur un corpus simulé que l'on nommera *train*. Ce modèle est ensuite évalué en utilisant la technique de validation croisée sur un corpus *test*.

Description des corpus. Chacun des jeux de données correspond à un ou plusieurs fichiers textes respectant le schéma suivant :

```
  Vocal_Cmd Usr_Loc Usr_Act -> Epxtd_Cmd Expctd_Loc

Exemples:
  blind - open kitchen none
         ->blind - open kitchen
  light - on kitchen cook
         ->light - on kitchen - sink
```

Nous nous concentrons pour le moment qu'à la prise de décision lorsque l'utilisateur interpelle le système par le biais d'une commande vocale. Ainsi, chacune des entrées des fichiers contient forcément un champ *Vocal_Cmd* valide de même que le champ *Usr_Loc* qui correspond à la localisation de l'utilisateur lorsqu'il prononce sa commande. En revanche, le champ *Usr_Act* qui symbolise l'activité courante de l'utilisateur peut prendre une valeur nulle (représentée dans notre cas par le mot-clé *none* du fait de notre implémentation en Python) qui signifie que l'on ne sait pas quelle activité est en train d'être effectuée.
À chacun des 324 états du système (qui ici correspondent au contexte courant) est alors associée une sortie escomptée, parmi les 32 possibles, composée de deux informations : la commande voulue, allumer la lumière, ouvrir le store et le lieu de l'action, le store de la chambre, la lumière de l'évier ou du plafond. Ces informations sont définies, pour le corpus *train*, par un ensemble de règles et dans le cas du corpus *test*, par un expert du domaine qui les a définies.
Le corpus *train* est un corpus qui doit nous permettre d'apprendre un premier modèle de notre environnement, notre modèle du monde. Pour cela, il doit être relativement important. Comme il n'existe que peu de corpus de données de décision, le corpus d'apprentissage a été généré de manière automatique en suivant la méthode employée par [3]. Nous utilisons pour cela un script qui parcourt chacun des états possibles de notre système et déduit la décision attendue d'un ensemble de règles logiques. De cette manière nous obtenons un corpus exhaustif représentant 380 interactions.
Le corpus *test* permet d'adapter et d'évaluer notre modèle. Il est donc important que ce corpus soit issu de données réelles. Nous avons donc choisi d'extraire les annotations des expérimentations réalisées lors du projet Sweet-Home [25]. Pour chacun des plis de la validation croisée, les données de 15 des 16 sujets sont utilisées pour l'apprentissage/adaptation du modèle et les données du dernier sujet sont utilisées pour l'évaluation. Chaque sujet réalisant un scénario similaire, nous considérons les données comme issues d'une seule et même personne répétant une même routine plusieurs fois. Les sujets ont réalisé en moyenne 25 interactions pour un total de 407 interactions recouvrant 37 des états possibles.

Déroulement de l'expérimentation. L'expérimentation s'est donc déroulée en 2 étapes : une phase d'entraînement

et une phase de validation. La phase de validation est elle-même divisée en 16 tours de validation croisée, chaque tour comprenant une phase d'adaptation du modèle et une phase d'évaluation. Chacune des phases est décomposée en 10 itérations à la fin desquelles les récompenses obtenues sont intégrées pour l'apprentissage.
Lors de l'entraînement nous utilisons le corpus *train* afin de simuler un grand nombre d'interactions, près de 100 000 (10 fois le produit des cardinalités des ensembles d'états et d'actions). On cherche de cette manière à s'assurer que tous les états aient été rencontrés plusieurs fois de façon à ce que le système ait pu explorer le résultat de différentes actions. Après cette phase nous obtenons une table de Q-values qui représente un modèle de l'environnement basé sur un ensemble de règles logiques.
À partir de ce modèle, nous utilisons une partie du corpus *test* pour adapter notre modèle du monde à la réalité terrain. L'adaptation se fait grâce à un ensemble de 375 interactions en moyenne qui est réutilisé 3 fois pour un total de 1125 interactions. À la fin de cette étape d'adaptation, nous obtenons un modèle prêt à être évalué sur le reste du corpus *test*.
Dans tous les cas, les règles d'apprentissages sont similaires. Un état est fourni au système qui retourne une action à appliquer. Cette action est comparée à l'action attendue. Si ces deux actions sont les mêmes alors le système est récompensé et un nouvel état lui est fourni en entrée. À l'inverse, si les deux actions sont différentes, une pénalité est infligée au système et ce dernier reste dans le même état. Dans le cas de l'apprentissage sur corpus simulé, le système à un nombre illimité d'essais avant de trouver l'action adéquate. En revanche, pour simuler l'impatience de l'utilisateur, lors de l'adaptation, le système n'a que trois essais, et enfin lors de l'évaluation le système n'a qu'une seule chance de prendre la bonne décision. À l'heure actuelle, la fonction de récompense est une fonction de *temps minimal à l'objectif* (*Minimum Time to Goal*) [13], toutefois il n'est pas exclu d'utiliser des fonctions moins classiques prenant en compte une similarité dans les actions ou pénalisant le système s'il met trop de temps à déterminer la bonne action.

Évaluation et métriques. D'une part, lors des phases d'entrainement et d'évaluation, nous gardons une trace de toutes les récompenses obtenues. Ceci nous permet de calculer la récompense moyenne obtenue par le système lors d'une séquence d'interactions, entre deux phases d'apprentissages. Dans le cas des données de tests, nous présentons uniquement l'évolution de la moyenne des récompenses au cours de l'adaptation du modèle. En effet, lors de l'évaluation, le système a pour but principal d'exploiter les connaissances acquises précédemment et il ne nous semblait alors moins pertinent d'étudier sa capacité d'apprentissage à ce moment-là.
D'autre part, notre système s'apparente ici grandement à un système de classification. De fait, il semble pertinent de présenter une matrice de confusion (normalisée sur les lignes) afin de rendre compte de la qualité de la classification.

3.3 Résultats

Suite à la réalisation du protocole ci-dessus, nous obtenons un graphique mettant en évidence la différence de récompense entre une première phase d'apprentissage et une seconde d'évaluation (Figure 4).

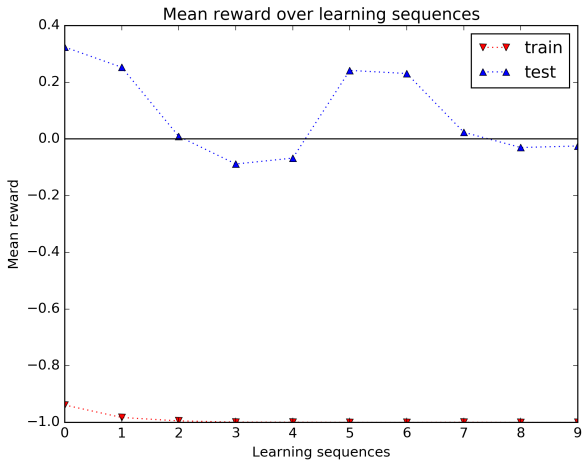

FIGURE 4 – Comparaison de la récompense moyenne lors des 10 séquences d'apprentissage

La mise en parallèle de ces courbes permet de mettre en exergue certaines particularités
Dans un premier temps, il y a une forte différence entre les deux phases. Cette différence peut s'expliquer par deux facteurs. D'une part, la phase d'apprentissage est plus exploratoire, le système va davantage testé de nouvelles actions, et si elles ne sont pas bonnes, il va réitérer jusqu'à trouver la bonne action. À l'inverse, lors de l'évaluation dès lors que le système a échoué 3 fois de suite, il passe à un nouvel état, ce qui limite sa perte de récompense. D'autre part, le nombre d'interactions par séquence diffère grandement entre les deux phases (de deux ordres de grandeurs). Ainsi sur 10 000 interactions, 20 récompenses positives auront un moins grand impact que sur 100 interactions.
En Figure 5 nous présentons la matrice de confusion issue de notre expérimentation. La diagonale nettement marquée de cette matrice atteste de la qualité de la classification réalisée par notre système, malgré quelques confusions entre les actions 19 et 20 (allumer le plafonnier de la cuisine / allumer la lampe de l'évier de la cuisine) et une mauvaise classification des actions 11 et 12 (extinction des lampes de la cuisine) qui semble au premier abord être dû à des erreurs d'annotation.

4 Discussion et perspectives

L'expérience reportée dans cet article, quoique restant assez artificielle, montre bien que les possibilités d'adaptation d'un système à un utilisateur sont possibles même

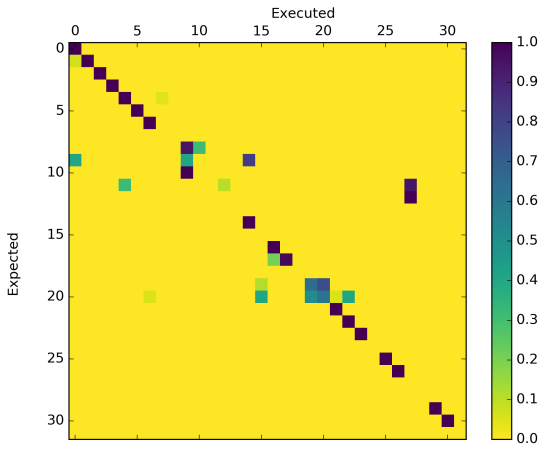

FIGURE 5 – Matrice de confusion pour la tâche de classification de contexte

avec une approche par renforcement aussi simple que le Q-learning. Cependant, un certain nombre de points n'ont pas été pris en compte. Les informations produites par le système domotique sont souvent utilisées directement par les systèmes de décision notamment pour de la régulation. Or, dans ce travail la décision est basée sur les informations de lieu et d'activité qui sont le résultat d'une inférence. Il convient donc de travailler sur l'estimation du transfert d'une récompense négative qui peut soit être due à une mauvaise décision soit à une inférence erronée. Pour cela, il convient de vérifier si des cadres théoriques prenant par nature l'incertitude en compte peuvent s'appliquer à ce double problème d'incertitude et de renforcement. Pour cela, les travaux de [11] dans lesquels un système à base de raisonnement flou s'adapte à la personne tout en prenant en compte l'incertitude sera étudié. Les Processus de décision markovien partiellement observable (POMDP) semblent également particulièrement adaptés à ce type de problème [28]. Enfin, il convient de vérifier si un système à base de MLN [3] peut être adapté pour inclure un mécanisme de renforcement. Un autre point important et non pris en compte dans l'étude est l'identification des personnes dans l'habitat. En effet, il convient d'associer un comportement à un utilisateur précis et de différencier les récompenses. Ce problème peut être inclus dans le problème précédent en considèrent une gamme restreinte d'utilisateur qu'il conviendrait d'identifier en même temps que le reste de l'environnement. Un autre point important est la validation d'une telle approche. Une adaptation sur le long terme nécessite une grande quantité de données qui est en pratique extrêmement coûteuse à mettre en place et pose des problèmes en termes de vérification (p.ex., différencier les adaptations essentielles des mineures). Notre stratégie sera d'une part de tester le système dans des nécessités d'adaptation radicale telle qu'avec un changement de personne ou d'habitat qui peuvent amener des informations sur la facilité d'« installation » du système dans un nouvel environnement. D'autre part, nous chercherons à tester une adaptation sur le long terme dans des environnements moins coûteux, potentiellement moins intrusif que l'habitat et avec peu de problème d'identification à travers des applications smart phone simples et des actions dans un bureau intelligent.

Références

[1] A. Badii and J. Boudy. CompanionAble - integrated cognitive assistive & domotic companion robotic systems for ability & security. In *1er Congrés of the Société Française des Technologies pour l'Autonomie et de Gérontechnologie (SFTAG'09)*, pages 18–20, Troyes, 2009.

[2] B. D. Carolis and G. Cozzolongo. C@sa : Intelligent home control and simulation. In *International Conference on Computational Intelligence*, pages 462–465, 2004.

[3] P. Chahuara, F. Portet, and M. Vacher. Making Context Aware Decision from Uncertain Information in a Smart Home : A Markov Logic Network Approach. In *Ambient Intelligence*, volume 8309 of *Lecture Notes in Computer Science*, pages 78–93, Dublin, Ireland, 2013. Springer.

[4] P. Chahuara, F. Portet, and M. Vacher. Making Context Aware Decision from Uncertain Information in a Smart Home : A Markov Logic Network Approach. In *Ambient Intelligence*, volume 8309 of *Lecture Notes in Computer Science*, pages 78–93, Dublin, Ireland, Dec. 2013. Springer.

[5] M. Chan, E. Campo, D. Estève, and J.-Y. Fourniols. Smart homes — current features and future perspectives. *Maturitas*, 64(2) :90–97, 2009.

[6] L. Cristoforetti, M. Ravanelli, M. Omologo, A. Sosi, A. Abad, M. Hagmueller, and P. Maragos. The DIRHA simulated corpus. In *The 9th edition of the Language Resources and Evaluation Conference (LREC)*, pages 2629–2634, Reykjavik, Iceland, 2014.

[7] G. Filho and T. J. Moir. From science fiction to science fact : a smart-house interface using speech technology and a photorealistic avatar. *International Journal of Computer Applications in Technology*, 39(8) :32–39, 2010.

[8] M. Gallissot, J. Caelen, F. Jambon, and B. Meillon. Une plateforme usage pour l'intégration de l'informatique ambiante dans l'habitat. l'appartement domus. *Technique et Science Informatiques (TSI)*, 32(5) :547–574, 2013.

[9] J. F. Gemmeke, B. Ons, N. Tessema, H. Van Hamme, J. Van De Loo, G. De Pauw, W. Daelemans, J. Huyghe, J. Derboven, L. Vuegen, B. Van Den Broeck, P. Karsmakers, and B. Vanrumste. Self-taught assistive vocal interfaces : an overview of the aladin project. In *Interspeech 2013*, pages 2039–2043, 2013.

[10] J. Gómez-Romero, M. A. Serrano, M. A. Patricio, J. García, and J. M. Molina. Context-based scene recognition from visual data in smart homes : an information fusion approach. *Personal and Ubiquitous Computing*, 16(7) :835–857, Oct. 2012.

[11] H. Hagras, F. Doctor, A. Lopez, and V. Callaghan. An incremental adaptive life long learning approach for type-2 fuzzy embedded agents in ambient intelligent environments. *IEEE Transactions on Fuzzy Systems*, 15(1) :41–55, 2007.

[12] M. Hamill, V. Young, J. Boger, and A. Mihailidis. Development of an automated speech recognition interface for personal emergency response systems. *Journal of NeuroEngineering and Rehabilitation*, 6, 2009.

[13] M. E. Harmon and S. S. Harmon. Reinforcement learning : A tutorial, 1996.

[14] D. Istrate, M. Vacher, and J.-F. Serignat. Embedded implementation of distress situation identification through sound analysis. *The Journal on Information Technology in Healthcare*, 6 :204–211, 2008.

[15] M. J. Kofler, C. Reinisch, and W. Kastner. A semantic representation of energy-related information in future smart homes. *Energy and Buildings*, 47 :169–179, Apr. 2012.

[16] S.-H. Lee and S.-B. Cho. Fusion of modular bayesian networks for context-aware decision making. In E. Corchado, V. Snášel, A. Abraham, M. Wozniak, M. Graña, and S.-B. Cho, editors, *Hybrid Artificial Intelligent Systems*, volume 7208 of *Lecture Notes in Computer Science*, pages 375–384. Springer Berlin / Heidelberg, 2012.

[17] C. Y. Leong, A. Ramli, and T. Perumal. A rule-based framework for heterogeneous subsystems management in smart home environment. *IEEE Transactions on Consumer Electronics*, 55(3) :1208–1213, 2009.

[18] K. Mitra, A. B. Zaslavsky, and C. Åhlund. A probabilistic context-aware approach for quality of experience measurement in pervasive systems. In *SAC*, pages 419–424, 2011.

[19] P. Moore, B. Hu, and M. Jackson. Rule strategies for intelligent context-aware systems : The application of conditional relationships in decision-support. In *International Conference on Complex, Intelligent and Software Intensive Systems, CISIS 2011*, pages 9–16, Seoul, Korea, 2011.

[20] M. C. Mozer. The Neural Network House : An Environment hat Adapts to its Inhabitants. In *Proc. AAAI Spring Symp. Intelligent Environments*, pages 110–114, 1998.

[21] T. Nishiyama, S. Hibiya, and T. Sawaragi. Development of agent system based on decision model for creating an ambient space. *AI & Society*, 26(3) :247–259, 2011.

[22] F. Portet, M. Vacher, C. Golanski, C. Roux, and B. Meillon. Design and evaluation of a smart home voice interface for the elderly : acceptability and objection aspects. *Personal and Ubiquitous Computing*, 17 :127–144, 2013.

[23] L. S. Shafti, P. A. Haya, M. García-Herranz, and E. Pérez. Inferring eca-based rules for ambient intelligence using evolutionary feature extraction. *Journal of Ambient Intelligence and Smart Environments*, 5(6) :563–587, 2013.

[24] M. Vacher, S. Caffiau, F. Portet, B. Meillon, C. Roux, E. Elias, B. Lecouteux, and P. Chahuara. Evaluation of a context-aware voice interface for Ambient Assisted Living : qualitative user study vs. quantitative system evaluation. *ACM Transactions on Accessible Computing*, 7(2) :1–36, 2015. (in press).

[25] M. Vacher, B. Lecouteux, P. Chahuara, F. Portet, B. Meillon, and N. Bonnefond. The Sweet-Home speech and multimodal corpus for home automation interaction. In *The 9th edition of the Language Resources and Evaluation Conference (LREC)*, pages 4499–4506, Reykjavik, Iceland, 2014.

[26] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

[27] S. S. Yau and J. Liu. Hierarchical situation modeling and reasoning for pervasive computing. In *Proceedings of the The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*, SEUS-WCCIA '06, pages 5–10, Washington, DC, USA, 2006. IEEE Computer Society.

[28] S. Zaidenberg and P. Reignier. Reinforcement Learning of User Preferences for a Ubiquitous Personal Assistant. In A. Mellouk, editor, *Advances in Reinforcement Learning*, pages 59–80. Intech, 2011.